FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

COMPUTER APPLACATION LAB

ENCS4110

Report#3

EXP.No.9.LCD interfacing with TM4C123

.....

Student Name: saja asfour

Student ID: 1210737

Instructor name: Abdel Salam Sayyad

Section number:2

Date: 20/6/2023

9.1 Abstract:

-**The Aim of the experiment:** learn to interface a 16*2 LCD with TM4C123 Tiva C Launchpad and learn about LCD components , how to write and simulate a program in LCD, and how to use push buttons to control movement direction .

Equipment Used in the experiment: keil vision 5 , TM4C123 Tiva Launchpad and 16*2 LCD.

Contents:

List of figure:

9.2 Theory

9.2.1 16*2 LCD Introduction:

*This LCD can display 32 ASCII characters

* It consists of two rows and one column.

* Each row can display one ASCII character.

* Hence , the position of each character is defined in terms of rows and column order pairs (x,y) .For example ,(0,0) means the first row and first column , and (1,15) means the seconds row and 15th column .

9.2.2 Function Description:

9.2.2.1 Registers:

- There are two 8-bit registers , an instruction registers (IR) , and data register on the HD44780U (DR) .

- the IR stores display clear and cursor shift instruction codes as well as address information for display data RAM (DDRAM) and character generator RAM (CGRAM).

- The IR can only be written from the MPU .

- The DR temporarily stores data to be written into DDRAM or CGRAM and temporarily stores data to read from DDRAM or CGRAM

- Data written into the DR from the MPU is automatically written into DDRAM or CGRAM by an internal operation .

-The DR is also used for data storage when reading data from DDRAM or CGRAM .

- when address information is written into the IR , data is read and then stored into the IR , data is read and then stored into the DR from DDRAM or CGRAM by an internal operation.

9.2.2.2 Memory:

In 16*2 LCD controller , there are three memory are available to store characters , numbers and special symbols . which are DDRAM (data display RAM) which stores ASCII codes, CGROM(character generating ROM) which is responsible for stored standard pattern, and CGRAM (character generating RAM) which holds customs character pattern space total 8 in 2x16 module.

9.2.2.2.1 Display Data RAM(DDRAM)

-Its stores display data represented in 8-bit character codes.

- its extended capacity is 80x8 bits, or 80 characters .

- the area in display data RAM(DDRAM) that is not used for display can be used as general data RAM.

9.2.2.2.2 character generator ROM(CGROM)

-that is responsible for stored standard character pattern generates 5x8 dot or 5x10 dot character patterns from 8-bit character codes

-it can generate 208 5x8 dot character patterns and 32 5x10 dot character patterns.

9.2.2.2.3 character generator RAM(CGRAM)

-The character generating RAM which holds custom character pattern has only 8 memory location available to store user defined characters with address 0x00 – 0x07 .

9.2.3 Displaying standard Character on LCD:

-out of these three memory locations, DDRAM and CGROM are used to generate regular standard characters (ASCII characters).

-By using these memory location , a user can generate different character fonts and symbols on LCD display . A character is designed by taking the number of pixels in mind .

-For example, in 16x2 LCD there are 16 segments available per single line .Each segment contains pixels in 5x7 or 5x10 matrix forms.

-For example , in 16x2 LCD there are 16 segments available per single line . Each segment contains pixels in 5x8 or 5x10 matrix form

-The DDRAM stores ASCII code of character which is sent by the microcontroller

9.2.4 pinout Diagram:

Figure 1: pinout diagram

9.2.4.1 16x2 pin Details

9.2.4.1.1 Data pin:

-pin D0 to D7 are data pin which used to send data to be displayed or commands to LCD Controller.

-hence , these lines will be connected with TM4C123 GPIO pins to transfer data .

-But this LCD can be used either in 4-bit (only D0 to D3 pins are used) or 8-bit mode(all D0 to D7 pins are used).

9.2.4.1.2 Control pins

-LCD contrast(v0): it used to adjust the contrast of LCD with respects to text display, This contrast can be set through by using a potential divider circuit with variable resistor.

-Register select (RS): with the help of this pin, TM4C123 microcontroller informs the LCD controller either we are sending commands or data to LCD . in other word, it helps to differentiate between data and commands.

-Read/Write (R/W) : as its name suggest, it used to select read or write mode of LCD, When this pin is set to active high , LCD will be in read mode which means we can read Data from the LCD . similarly , when pin is set to active low, we will be able to write data to The LCD , we will connect this pin to the ground because we are using a 16x2 LCD as an Output device only.

-Enable(E): This pin used to enable and disable LCD . when this pin is active low , LCD Controller will be disabled . That means control pins and data pins will not have any effect on the display . On the other hand, when the enable pin is set to active high , the LCD will work normally and process all data and control instructions.

-BackLight LED pins: These pin are cathode and anode pins of back light LED , they Are used to provide +5 volts and ground to anode and cathode pins.

-Power supply pins: +5 volt power supply pins.

9.2.5 Difference Between 4-bit and 8-bit mode:

HD47780 LCDs can be interfaced with TM4C123 Tiva Launchpad either in 4- bit or 8-bit mode .

- For a 4-bit interface , we need to use 6 GPIO pins of Launchpad.

-in 4 bit mode, data transfers from microcontroller to the LCD in two consecutive half bytes.

-For 8-bit mode , one byte data transfers to the LCD in one go

9.3 Procedure

9.3.1 LCD interfacing with TM4C123 Tiva Launchpad in 4-bit Mode :

we use a 4-bit mode to interface 16x2 LCD with TM4C123 microcontroller , so we make the connection with 16x2 Lcd with TM4C123 according to this digram:

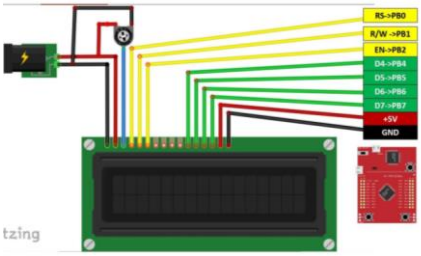


Figure 2: connection diagram

9.3.2 Example :

This code displays "ENCS4110" on the first row an "LAB" on the second Row of the LCD.

9.3.2.1 Example code:

```
#include "TM4C123.h" // Device header
#define LCD GPIOB //LCD port with Tiva C
#define RS 0x01 //RS -> PB0 (0x01)
#define RW 0x02 //RW -> PB1 (0x02)
#define EN 0x04 //EN -> PB2 (0x04)
//Functions Declaration
void delayUs(int); //Delay in Micro Seconds
void delayMs(int); //Delay in Milli Seconds
void LCD4bits_Init(void); //Initialization of LCD Dispaly
void LCD_Write4bits(unsigned char, unsigned char); //Write data as (4 bits) on LCD
void LCD_WriteString(char*); //Write a string on LCD
void LCD4bits_Cmd(unsigned char); //Write command
void LCD4bits_Data(unsigned char); //Write a character
int main(void)
{
char* str1 = "ENCS4110…"; //Write any string you want to display on the first row of LCD
 char* str2 = "Lab …"; //Write any string you want to display on the second row of LCD
 LCD4bits_Init(); //Initialization of LCD
 delayMs(500); //delay 500ms for LCD initialization
LCD4bits_Cmd(0x01); //Clear the display
 delayMs(500);
LCD4bits_Cmd(0x80); //Force the cursor to beginning of 1st line
delayMs(500); //delay 500ms for LCD (MCU is faster than LCD)
LCD_WriteString(str1); //Write the string on LCD
delayMs(500); //Delay 500 ms to let the LCD diplays the data
 LCD4bits_Cmd(0xC0); //Force the cursor to beginning of 2
nd line
delayMs(500); //delay 500ms for LCD (MCU is faster than LCD)
LCD_WriteString(str2); //Write the string on LCD
```

```
LCD_WriteString(str2); //Write the string on LCD
delayMs(500); //Delay 500 ms to let the LCD diplays the data
}
11
void LCD4bits_Init(void)
{
SYSCTL->RCGCGPIO |= 0x02; //enable clock for PORTB
delayMs(10); //delay 10 ms for enable the clock of PORTB
 LCD->DIR = 0xFF; //let PORTB as output pins
LCD->DEN = 0xFF; //enable PORTB digital IO pins
LCD4bits_Cmd(0x28); //2 lines and 5x7 character (4-bit data, D4 to D7)
 delayMs(100);
LCD4bits_Cmd(0x06); //Automatic Increment cursor (shift cursor to right)
 delayMs(100);
LCD4bits_Cmd(0x01); //Clear display screen
 delayMs(100);
LCD4bits_Cmd(0x0F); //Display on, cursor blinking
}
void LCD_Write4bits(unsigned char data, unsigned char control)
{
data &= 0xF0; //clear lower nibble for control
control &= 0x0F; //clear upper nibble for data
LCD->DATA = data | control; //Include RS value (command or data ) with data
LCD->DATA = data | control | EN; //pulse EN
delayUs(10); //delay for pulsing EN
LCD->DATA = data | control; //Turn off the pulse EN
LCD->DATA = 0; //Clear the Data
}
void LCD_WriteString(char * str)
```

```
56  }
57  void LCD_WriteString(char * str)
58  {
59  volatile int i = 0; //volatile is important
60  while(*(str+i) != '\0') //until the end of the string
61  {
62  LCD4bits_Data(*(str+i)); //Write each character of string
63  i++; //increment for next character
64  }
65  }
66  void LCD4bits_Cmd(unsigned char command)
67  {
68  LCD_Write4bits(command & 0xF0 , 0); //upper nibble first
69  LCD_Write4bits(command << 4 , 0); //then lower nibble
70  if(command < 4)
71  delayMs(2); //commands 1 and 2 need up to 1.64ms
72  else
73  delayUs(40); //all others 40 us
74  }
75  void LCD4bits_Data(unsigned char data)
76  {
77  LCD_Write4bits(data & 0xF0 , RS); //upper nibble first
78  LCD_Write4bits(data << 4 , RS); //then lower nibble
79  delayUs(40); //delay for LCD (MCU is faster than LCD)
80  }
81  void delayMs(int n)
82  {
83  volatile int i,j; //volatile is important for variables incremented in code
84  for(i=0;i<n;i++)
```

```
66  void LCD4bits_Cmd(unsigned char command)
67  {
68  LCD_Write4bits(command & 0xF0 , 0); //upper nibble first
69  LCD_Write4bits(command << 4 , 0); //then lower nibble
70  if(command < 4)
71  delayMs(2); //commands 1 and 2 need up to 1.64ms
72  else
73  delayUs(40); //all others 40 us
74  }
75  void LCD4bits_Data(unsigned char data)
76  {
77  LCD_Write4bits(data & 0xF0 , RS); //upper nibble first
78  LCD_Write4bits(data << 4 , RS); //then lower nibble
79  delayUs(40); //delay for LCD (MCU is faster than LCD)
80  }
81  void delayMs(int n)
82  {
83  volatile int i,j; //volatile is important for variables incremented in code
84  for(i=0;i<n;i++)
85  for(j=0;j<3180;j++) //delay for 1 msec
86  {}
87  }
88  void delayUs(int n)
89  {
90  volatile int i,j; //volatile is important for variables incremented in code
91  for(i=0;i<n;i++)
92  for(j=0;j<3;j++) //delay for 1 micro second
93  {}
94  }
```

Figure 3: example code

9.3.2.2 Example Discussion:

-from line 75 to line 80 : we give an active high to low transition pulse to enable pin to perfome a write operation on the LCD . finally, send nubble to LCD.

-from line 66 to 74: This function used to send commands to LCD . all commands are of 8-bits . Hence , first this routine sends the upper nibble of command to LCD and after that it sends the lower nibble . we need to add delay between sending commands . command 1 and 2 takes 1.64ms and all other commands take 40us . therefore, we add a delay of 2ms if commands are less than 4 and 40us for all other commands.

-from line 33 to line 46: this routine enables the clock to TM4C123 GPIOB which is connected to control and data pins of 16x22 LCD . After we send various initialization commands to LCD from control pins of TM4C123 microcontroller such as :

-set the character font size to 5x7which 5 represents the number of rows and 7 the number Of columns

-select 4-bit Mode to transfer data or command in two nibbles

-setting to move cursor posiition right after displaying each charcter

-Clear the screen

- Enable light and Cursor blinking

-from line 47 to line 56: This function to prints charcter on the current cursor position of the display . Acharacter type of data consist of 8-bits . But we are using 4-bit mode of LCD. Hence , data transfers in two nibbles or 2 pieces of 4-bits by using LCD4bits_Data() function.The one importamt point to note here is that we are sending control signal RS which will be defined as an active high micro in the code because we set RS signal to active high when we want to transmit data from TM4C123 Tiva Launchpad to LCD.

-from line 57 to line 65: this to write string to LCD starting from current cursor position . it use LCD4bits_Data() function to send all characters of a string one by one . and this function keeps writing characters till the NULL character is found.

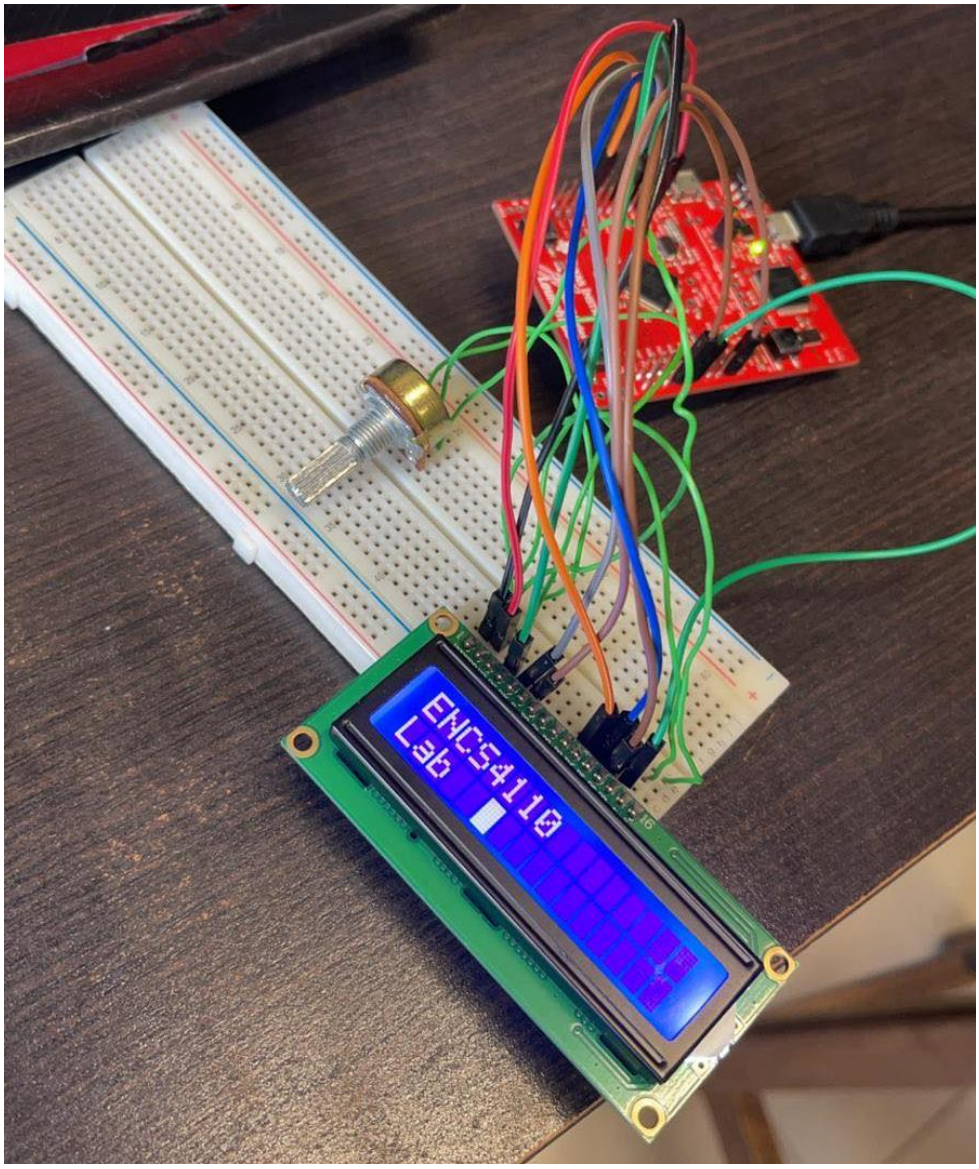


Figure 4: the result of example

9.4 Lab Work:

9.4.1 Question 1:

Modify the code to display your name on the first line and your ID on the second line:

I just modify the strings in main function

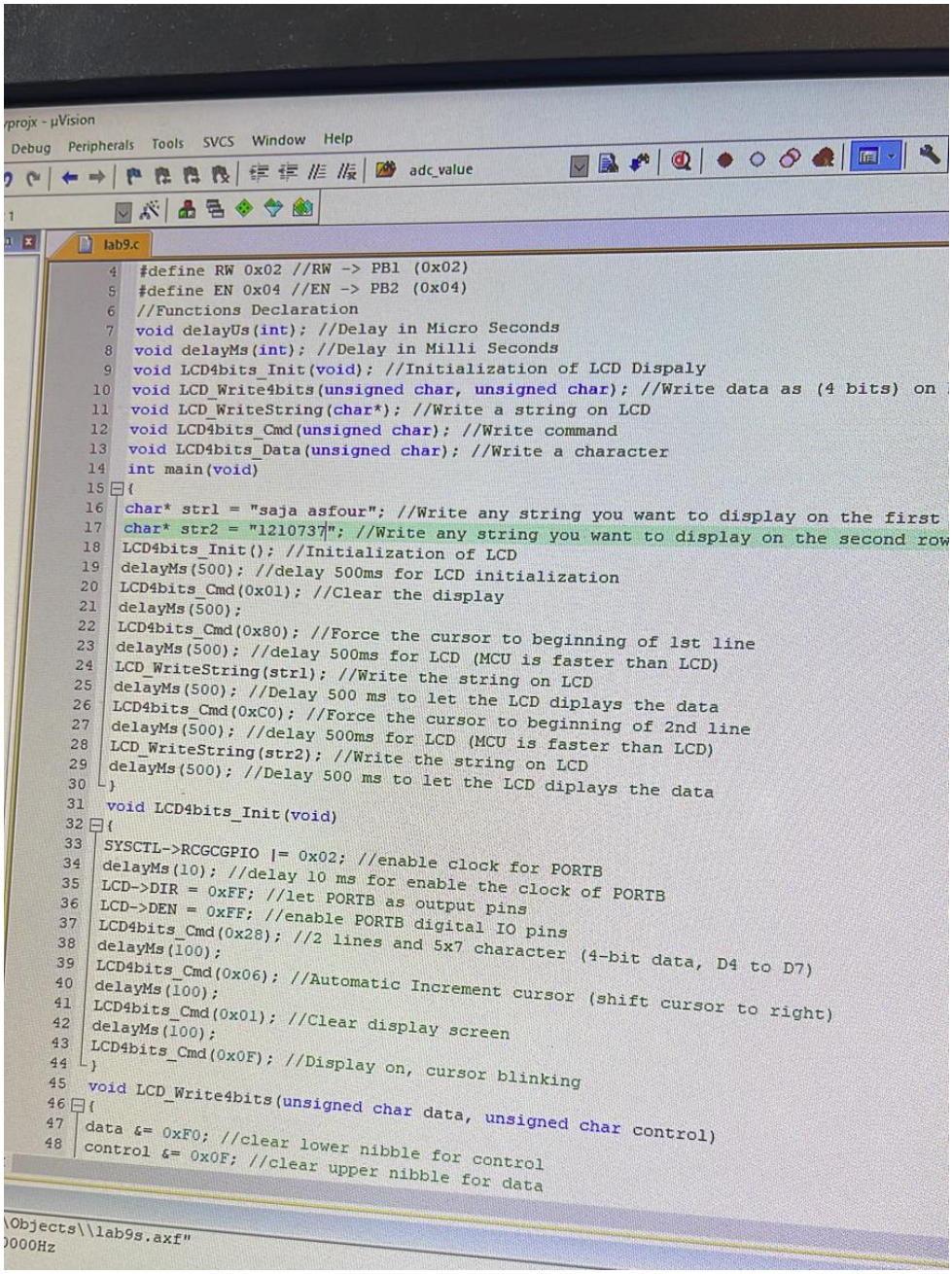


Figure 5: task 1 code

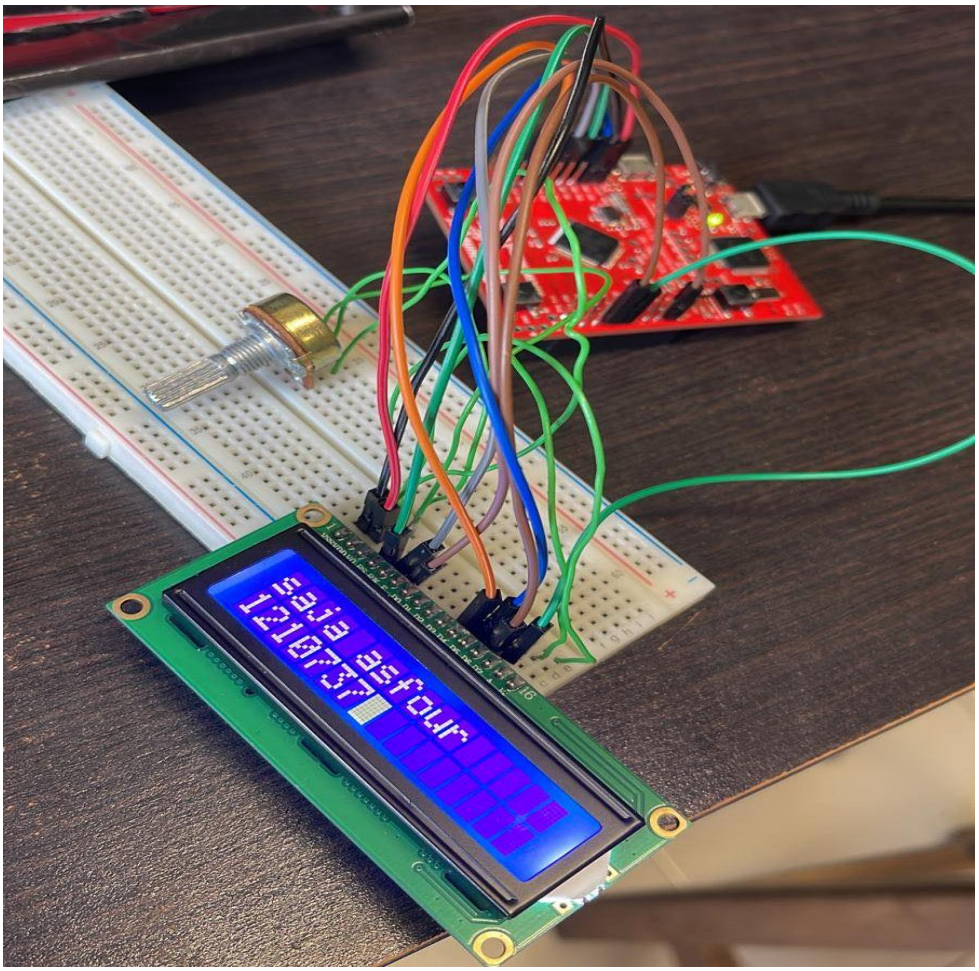


Figure 6: task 1 result

9.4.2 Question 2:

Write a program that display your name on LCD with movement . Your program should allow the user to control the direction of the movement (shift left, shift right) using the two on-bored push buttons .

I add the handler function to use SW1 and SW2 , and make a variable a which consist 0x80 which mean beginning of the first line , then in main function I make a while loop to write when I push the buttons

lab9.c

```c
#include "TM4C123.h" // Device header
#define LCD GPIOB //LCD port with Tiva C
#define RS 0x01 //RS -> PB0 (0x01)
#define RW 0x02 //RW -> PB1 (0x02)
#define EN 0x04 //EN -> PB2 (0x04)
//Functions Declaration
void delayUs(int); //Delay in Micro Seconds
void delayMs(int); //Delay in Milli Seconds
void LCD4bits_Init(void); //Initialization of LCD Dispaly
void LCD_Write4bits(unsigned char, unsigned char); //Write data as (4 bits) on LCD
void LCD_WriteString(char*); //Write a string on LCD
void LCD4bits_Cmd(unsigned char); //Write command
void LCD4bits_Data(unsigned char); //Write a character
int a=0x80;
void GPIOF_Handler(void);
int main(void)
{
  char* strl = "saja asfour"; //Write any string you want to display on the first row of LCD
  SYSCTL->RCGCGPIO |= (1<<5); /* Set bit5 of RCGCGPIO to enable clock to PORTF*/
/* PORTF0 has special function, need to unlock to modify */
GPIOF->LOCK = 0x4C4F434B; /* unlock commit register */
GPIOF->CR = 0x01; /* make PORTF0 configurable */
GPIOF->LOCK = 0; /* lock commit register */
/*Initialize PF3 as a digital output, PF0 and PF4 as digital input pins */
GPIOF->DIR &= ~(1<<4)|~(1<<0); /* Set PF4 and PF0 as a digital input pins */
GPIOF->DIR |= (1<<3); /* Set PF3 as digital output to control green LED */
GPIOF->DEN |= (1<<4)|(1<<3)|(1<<0); /* make PORTF4-0 digital pins */
GPIOF->PUR |= (1<<4)|(1<<0); /* enable pull up for PORTF4, 0 */
/* configure PORTF4, 0 for falling edge trigger interrupt */
GPIOF->IS &= ~(1<<4)|~(1<<0); /* make bit 4, 0 edge sensitive */
GPIOF->IBE &=~(1<<4)|~(1<<0); /* trigger is controlled by IEV */
GPIOF->IEV &= ~(1<<4)|~(1<<0); /* falling edge trigger */
GPIOF->ICR |= (1<<4)|(1<<0); /* clear any prior interrupt */
GPIOF->IM |= (1<<4)|(1<<0); /* unmask interrupt */
/* enable interrupt in NVIC and set priority to 3 */
NVIC->IP[30] = 3 << 5; /* set interrupt priority to 3 */
NVIC->ISER[0] |= (1<<30); /* enable IRQ30 (D30 of ISER[0]) */



LCD4bits_Init(); //Initialization of LCD
delayMs(500); //delay 500ms for LCD initialization

while(1){
LCD4bits_Cmd(0x01); //Clear the display
```

```c

while(1){
 LCD4bits_Cmd(0x01); //Clear the display
 delayMs(500);
 LCD4bits_Cmd(a); //Force the cursor to beginning of 1st line
 delayMs(500); //delay 500ms for LCD (MCU is faster than LCD)
 LCD_WriteString(strl); //Write the string on LCD
 delayMs(500); //Delay 500 ms to let the LCD diplays the data




}
}
 void LCD4bits_Init(void)
{
 SYSCTL->RCGCGPIO |= 0x02; //enable clock for PORTB
 delayMs(10); //delay 10 ms for enable the clock of PORTB
 LCD->DIR = 0xFF; //let PORTB as output pins
 LCD->DEN = 0xFF; //enable PORTB digital IO pins
 LCD4bits_Cmd(0x28); //2 lines and 5x7 character (4-bit data, D4 to D7)
 delayMs(100);
 LCD4bits_Cmd(0x06); //Automatic Increment cursor (shift cursor to right)
 delayMs(100);
 LCD4bits_Cmd(0x01); //Clear display screen
 delayMs(100);
 LCD4bits_Cmd(0x0F); //Display on, cursor blinking
}
 void LCD_Write4bits(unsigned char data, unsigned char control)
{
 data &= 0xF0; //clear lower nibble for control
 control &= 0x0F; //clear upper nibble for data
 LCD->DATA = data | control; //Include RS value (command or data ) with data
 LCD->DATA = data | control | EN; //pulse EN
 delayUs(10); //delay for pulsing EN
 LCD->DATA = data | control; //Turn off the pulse EN
 LCD->DATA = 0; //Clear the Data
}
 void LCD_WriteString(char * str)
{
 volatile int i = 0; //volatile is important
 while(*(str+i) != '\0') //until the end of the string
{
 LCD4bits_Data(*(str+i)); //Write each character of string
 i++; //increment for next character
}
```

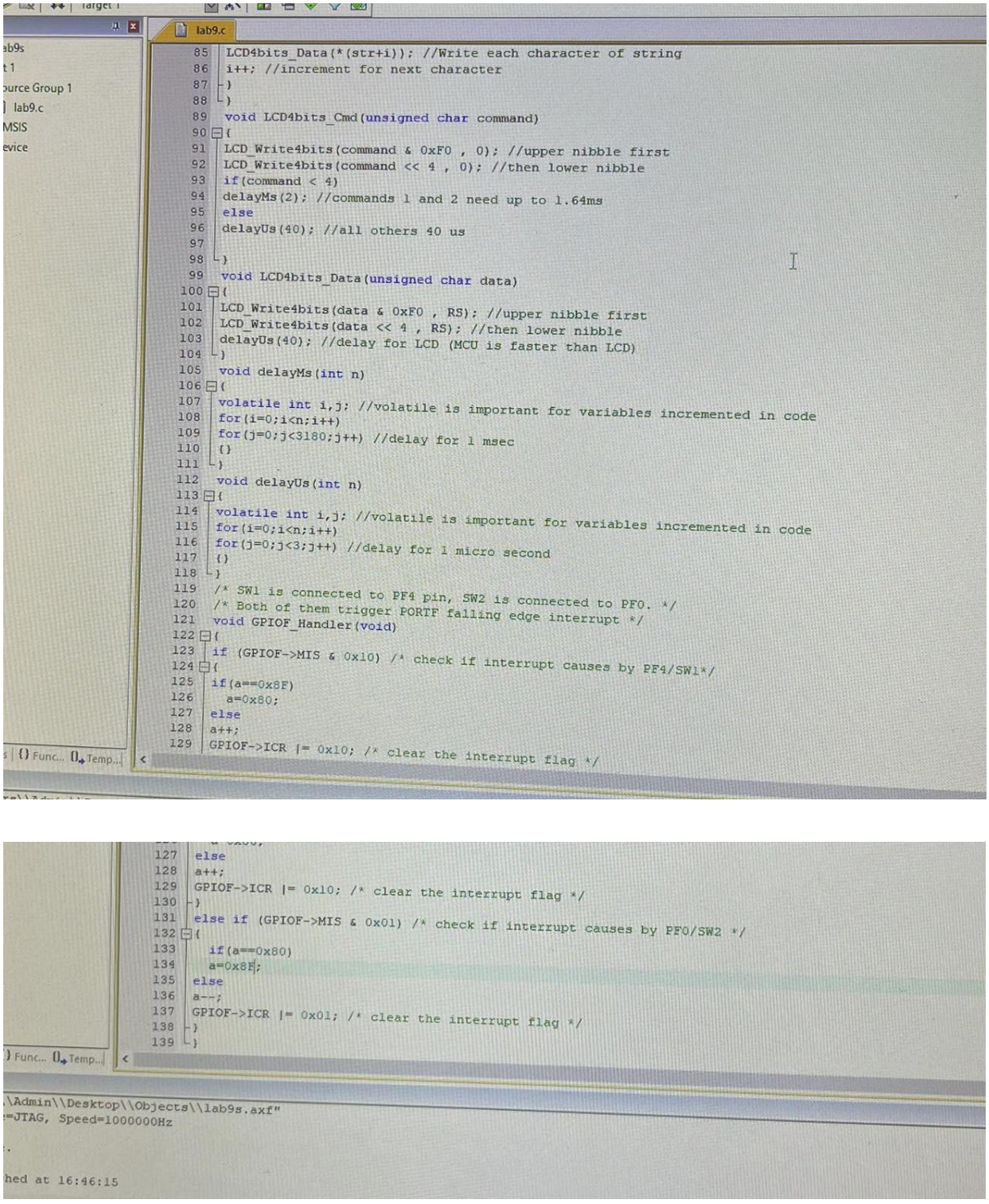
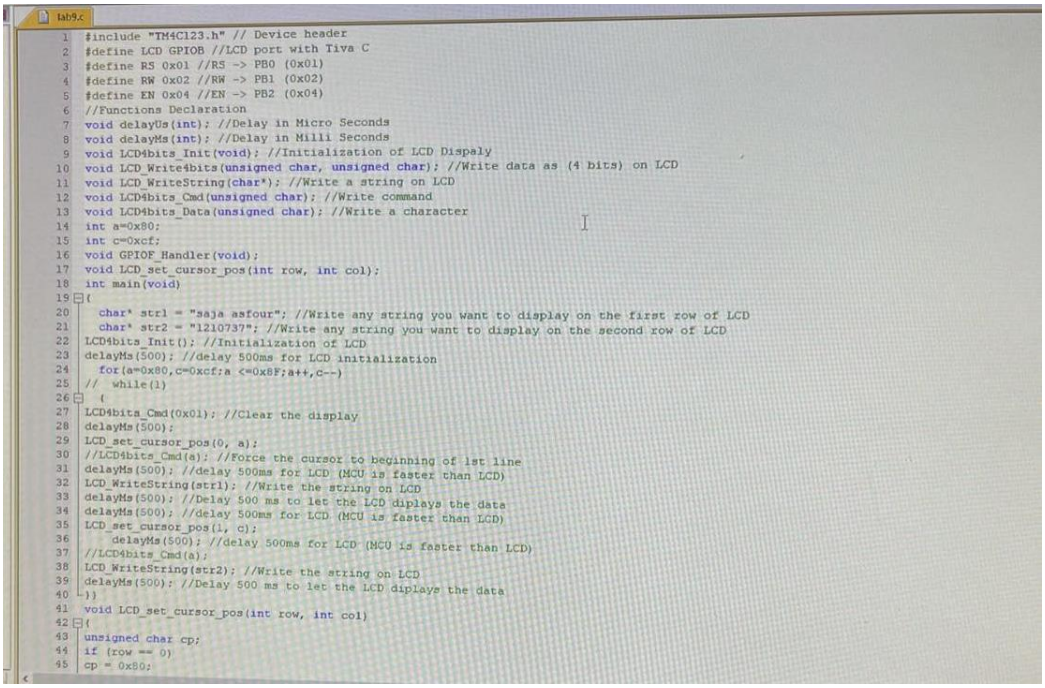


Figure 7 :task 2 code

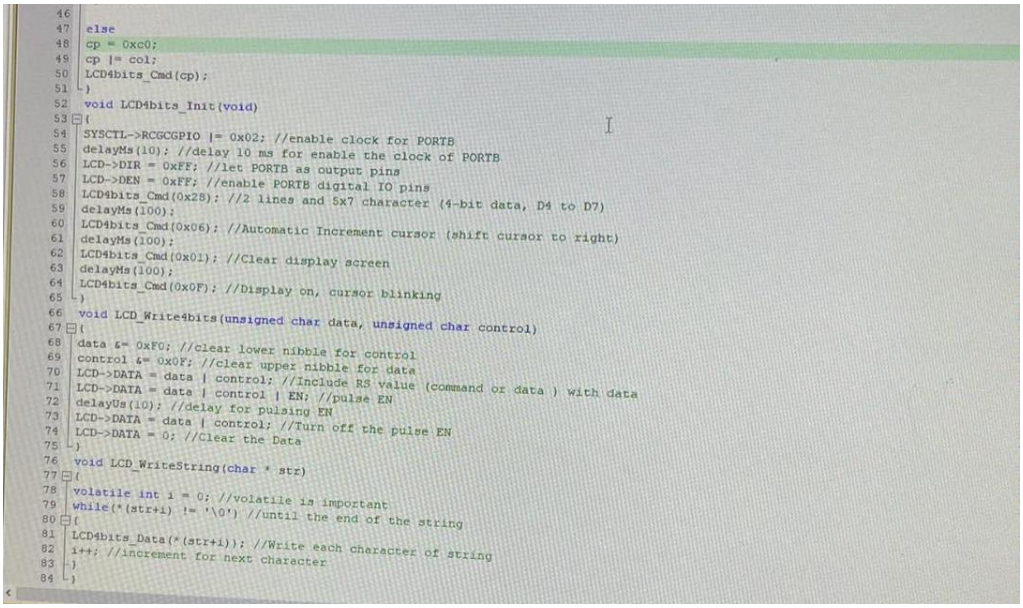
9.4.3 Question 3:

Write a program that display your name in the first row and your Id in the second row on the LCD. The Upper word should be firstly appears from the left of the LCD then it shifted continually to the other side . The lower word must have the opposite movement at the time . All that happen after a button press from the user .

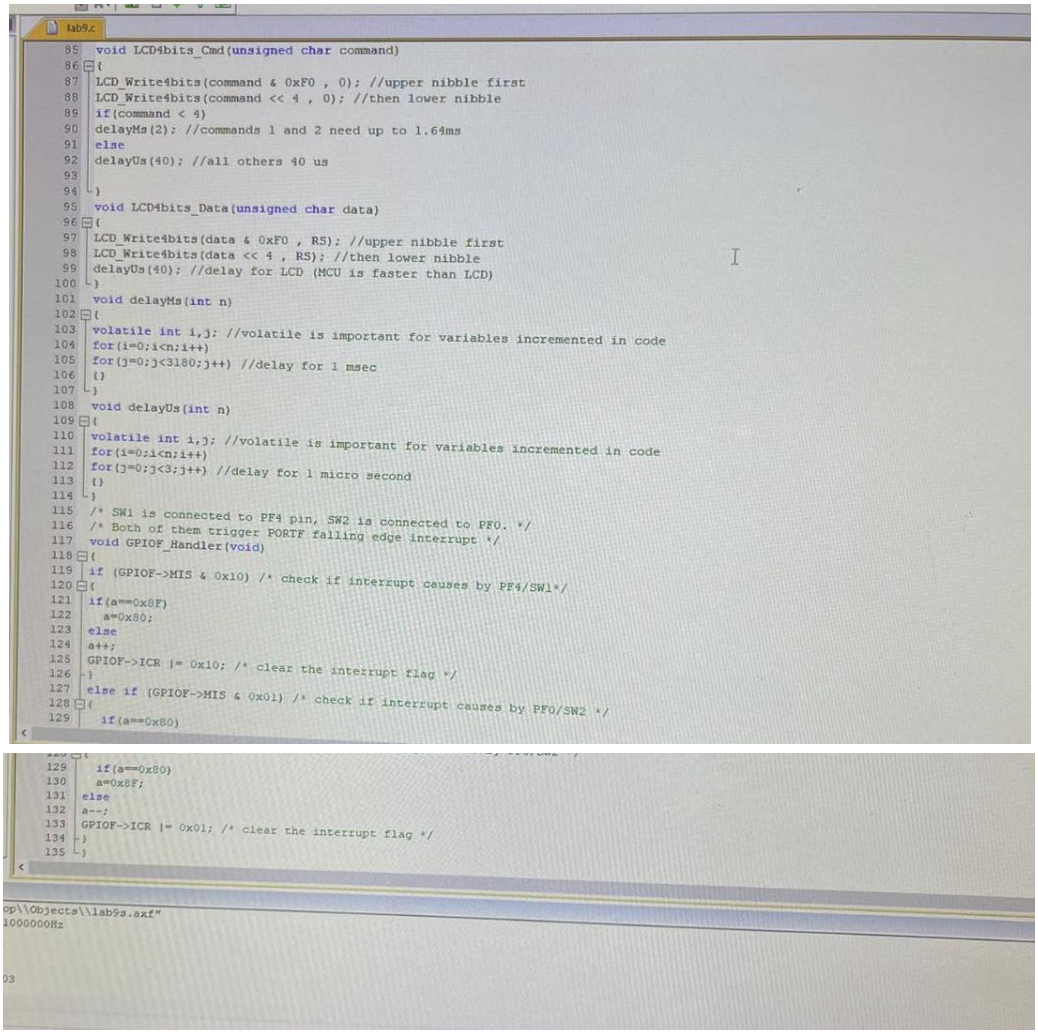
-Here I define two integer one is for the first line and the other for the second line and make a for loop to control the shifted



```
lab9.c
#include "TM4C123.h" // Device header
#define LCD_GPIOB //LCD port with Tiva C
#define RS 0x01 //RS -> PB0 (0x01)
#define RW 0x02 //RW -> PB1 (0x02)
#define EN 0x04 //EN -> PB2 (0x04)
//Functions Declaration
void delayUs(int); //Delay in Micro Seconds
void delayMs(int); //Delay in Milli Seconds
void LCD4bits_Init(void); //Initialization of LCD Dispaly
void LCD_Write4bits(unsigned char, unsigned char); //Write data as (4 bits) on LCD
void LCD_WriteString(char*); //Write a string on LCD
void LCD4bits_Cmd(unsigned char); //Write command
void LCD4bits_Data(unsigned char); //Write a character
int a=0x80;
int c=0xcf;
void GPIOF_Handler(void);
void LCD_set_cursor_pos(int row, int col);
int main(void)
{
  char* str1 = "saja asfour"; //Write any string you want to display on the first row of LCD
  char* str2 = "1210737"; //Write any string you want to display on the second row of LCD
LCD4bits_Init(); //Initialization of LCD
delayMs(500); //delay 500ms for LCD initialization
  for(a=0x80,c=0xcf;a <=0x8F;a++,c--)
//  while(1)
  {
LCD4bits_Cmd(0x01); //Clear the display
delayMs(500);
LCD_set_cursor_pos(0, a);
//LCD4bits_Cmd(a); //Force the cursor to beginning of 1st line
delayMs(500); //delay 500ms for LCD (MCU is faster than LCD)
LCD_WriteString(str1); //Write the string on LCD
delayMs(500); //Delay 500 ms to let the LCD diplays the data
delayMs(500); //delay 500ms for LCD (MCU is faster than LCD)
LCD_set_cursor_pos(1, c);
    delayMs(500); //delay 500ms for LCD (MCU is faster than LCD)
//LCD4bits_Cmd(a);
LCD_WriteString(str2); //Write the string on LCD
delayMs(500); //Delay 500 ms to let the LCD diplays the data
}}
void LCD_set_cursor_pos(int row, int col)
{
unsigned char cp;
if (row == 0)
cp = 0x80;
```



```
else
cp = 0xc0;
cp |= col;
LCD4bits_Cmd(cp);
}
 void LCD4bits_Init(void)
{
 SYSCTL->RCGCGPIO |= 0x02; //enable clock for PORTB
 delayMs(10); //delay 10 ms for enable the clock of PORTB
 LCD->DIR = 0xFF; //let PORTB as output pins
 LCD->DEN = 0xFF; //enable PORTB digital IO pins
 LCD4bits_Cmd(0x28); //2 lines and 5x7 character (4-bit data, D4 to D7)
 delayMs(100);
 LCD4bits_Cmd(0x06); //Automatic Increment cursor (shift cursor to right)
 delayMs(100);
 LCD4bits_Cmd(0x01); //Clear display screen
 delayMs(100);
 LCD4bits_Cmd(0x0F); //Display on, cursor blinking
}
 void LCD_Write4bits(unsigned char data, unsigned char control)
{
 data &= 0xF0; //clear lower nibble for control
 control &= 0x0F; //clear upper nibble for data
 LCD->DATA = data | control; //Include RS value (command or data ) with data
 delayUs(10); //delay for pulsing EN
 LCD->DATA = data | control | EN; //pulse EN
 LCD->DATA = data | control; //Turn off the pulse EN
 LCD->DATA = 0; //Clear the Data
}
 void LCD_WriteString(char * str)
{
 volatile int i = 0; //volatile is important
 while(*(str+i) != '\0') //until the end of the string
{
 LCD4bits_Data(*(str+i)); //Write each character of string
 i++; //increment for next character
}
}
```



```
lab9.c
85  void LCD4bits_Cmd(unsigned char command)
86  {
87  LCD_Write4bits(command & 0xF0 , 0); //upper nibble first
88  LCD_Write4bits(command << 4 , 0); //then lower nibble
89  if(command < 4)
90  delayMs(2); //commands 1 and 2 need up to 1.64ms
91  else
92  delayUs(40); //all others 40 us
93
94  }
95  void LCD4bits_Data(unsigned char data)
96  {
97  LCD_Write4bits(data & 0xF0 , RS); //upper nibble first
98  LCD_Write4bits(data << 4 , RS); //then lower nibble
99  delayUs(40); //delay for LCD (MCU is faster than LCD)
100 }
101 void delayMs(int n)
102 {
103 volatile int i,j; //volatile is important for variables incremented in code
104 for(i=0;i<n;i++)
105 for(j=0;j<3180;j++) //delay for 1 msec
106 {}
107 }
108 void delayUs(int n)
109 {
110 volatile int i,j; //volatile is important for variables incremented in code
111 for(i=0;i<n;i++)
112 for(j=0;j<3;j++) //delay for 1 micro second
113 {}
114 }
115 /* SW1 is connected to PF4 pin, SW2 is connected to PF0. */
116 /* Both of them trigger PORTF falling edge interrupt */
117 void GPIOF_Handler(void)
118 {
119 if (GPIOF->MIS & 0x10) /* check if interrupt causes by PF4/SW1*/
120 {
121 if(a==0x8F)
122   a=0x80;
123 else
124 a++;
125 GPIOF->ICR |= 0x10; /* clear the interrupt flag */
126 }
127 else if (GPIOF->MIS & 0x01) /* check if interrupt causes by PF0/SW2 */
128 {
129   if(a==0x80)
130   a=0x8F;
131 else
132 a--;
133 GPIOF->ICR |= 0x01; /* clear the interrupt flag */
134 }
135 }
```

```
op\\Objects\\lab9s.axf"
1000000Hz

03
```

Figure 8: task 3 code

9.5 Conclusion

After completing this experiment , we learn how to use LCD components , write and simulate a program in LCD , and control the movement direction.

9.6 References

- https://ritaj.birzeit.edu/bzu-msgs/attach/2297095/Exp9_LCD_Interface.pdf